Exercise 1. (*Platooning with adaptive cruise control*)
Your goal in this exercise is to have 5 vehicles drive in a platoon formation. Use the path tracking mode of the vehicles for this exercise and further exercises. The main control goals in platooning using ACC can be formulated by the following requirements:

1. Stability of the controlled vehicles
2. Synchronization: $\lim_{t\to\infty} |v_i(t) - v_0(t)| = 0, \quad i = 1, \ldots, n_{\text{veh}}$
3. Time-headway policy: $d_{i\text{ref}}(t) = T_{i,d}\, v_i(t) + d_{i\,\min}, \quad i = 1, \ldots, n_{\text{veh}}$
4. Collision avoidance: $d_i(t) \geq d_{i\,\min}, \quad t \geq 0, i = 1, \ldots, n_{\text{veh}}$
5. Continuous progression: $v_i(t) \geq 0, \quad t \geq 0, i = 0, \ldots, n_{\text{veh}}$

Here, the index i is used for the i-th agent in the platoon, the index 0 is used for the leader. Usually, the minimum distance is measured between front and rear bumper of consecutive vehicles. Since our position system provides the position of the center of gravity (CG) of the vehicles, for simplicity d_i describes the distance between the CG of vehicle i to the CG of vehicle $i-1$ along the path they are following.

For a sound PID design, you would have to show string stability as a necessary condition and a further sufficient condition for collision avoidance. This is beyond the scope of this course, but you can find more on that in [1].

a) Choose a communication structure for the networked control system (NCS) and reason about your choice. Draw the resulting graph representing the agent interaction you designed.

b) The function `compute_distance_on_path` that you used in the last exercise determines the distance along the path in a brute force way. When controlling multiple vehicles, you will be short on time in each control loop. Improve the function so that it works more efficiently on the 'outer_lane' path, but stick to the resolution of at least 1 cm.

c) Design a PID-controller for each agent for ACC. Can you tune your controller to meet the requirements of platooning? What minimum time-headway can you achieve without collisions and for how many vehicles? Can you guarantee stability without the time-headway, i.e., with a constant reference distance? Test your system for its limits with step changes in the leader's velocity.

Exercise 2. (*Centralized platoon control*)
Your goal in this exercise is to realize a platoon using centralized model predictive control (CMPC).

Have a look at the case study examples given in the lecture for ideas on how to model the system.

a) Choose a communication structure for the NCS and reason about your choice. Draw the resulting graph representing the agent interaction you designed.

b) Find a suitable state-space representation of the NCS consisting of all agents.

c) Implement a CMPC for the system. Aim to maintain a constant reference distance d_{ref} between the CGs of consecutive vehicles of

$$d_{\text{ref}} = 0.5\,\text{m}. \tag{1}$$

Apply the following state constraint on the distance and the input and change of input constraints:

$$\begin{aligned} d_{\text{min}} &= 0.3\,\text{m} & v_{\text{min}} &= 0\,\text{m/s} & a_{\text{min}} &= -1\,\text{m/s}^2 \\ & & v_{\text{max}} &= 1.5\,\text{m/s} & a_{\text{max}} &= 0.5\,\text{m/s}^2 \end{aligned} \tag{2}$$

If quadprog gets stuck in infeasible regions, think about possible reasons. Implement a slack variable for the distance in the class `ModelPredictiveControl`. Have it appear in the linear term of the cost function with a high weight (e.g. 10^6) and constrain it to be positive.

Exercise 3. (*Distributed platoon control*)
Your goal in this exercise is to realize a platoon using distributed model predictive control (DMPC).

a) Find a suitable state-space representation of each agent in the platoon.

b) Choose a communication structure for the NCS and reason about your choice. Draw the resulting graph representing the agent interaction you designed. Which agents can compute their control actions in parallel, which in sequence?

c) Define a messages to exchange the predicted output trajectories between the high-level controllers with the interface definition languageformat. Create a test program with a data writer and a data reader to test your communication (see `high_level_controller/examples/matlab/init_script.m` for an example).

d) Implement a DMPC controller. Use the same reference as in Equation 1 and the same constraints as in Equation 2. Communicate through the message format you created in Task c). Your goal is to achieve the best performance, so use any information allowed by your control strategy and communication structure that helps you in that regard.

We would be grateful if you evaluated this lab exercise. Please take the time to fill out the form you have received per email to help us improve this course. We are especially happy to read what you wrote in the text field, e.g. what you liked, what you haven't liked and how it could be improved.

Exercise 4. (*Exam preparation*)
Evaluate the DMPC controller developed in the exercise Exercise 3 to show its performance in the exam. Again, use the same reference as in Equation 1 and the same constraints as in Equation 2.

Evaluate your implementation with the following scenario: Vehicles with IDs 1, 2, 3, 5 and 7 form a platoon. Vehicle 7 is the leading vehicles, while the others follow in descending order. The vehicles start in distances unequal to their reference distance as given through the initial poses of the simulated vehicles.

The leader should now follow the speed profile given by

$$v_{0\text{ref}} = \begin{cases} 0.5\,\text{m/s}, & 0\,\text{s} \leq t < 15\,\text{s} \\ 1.4\,\text{m/s}, & 15\,\text{s} \leq t < 25\,\text{s} \\ 0.8\,\text{m/s}, & 25\,\text{s} \leq t < 35\,\text{s} \\ 0.0\,\text{m/s}, & 35\,\text{s} \leq t. \end{cases} \tag{3}$$

The followers should reach the leader's velocity and the reference distance d_{ref} to their predecessor. Stop the experiment at $t = 40\,\text{s}$.

If the situation allows it, we will run your DMPC controller in the lab. The first line of `TEAMREPO/23_distributed_platoon_control/main_dmpc.m` must be a comment that states the necessary sample time setting for the middleware. Push your code before the deadline given in moodle. The most recent version of your code before the deadline will be used in the experiments.

We will provide you with plots (the corresponding function will also be added to the template repo) and a video showing your distributed control algorithm in the lab. After we have provided the experimental results, download the slide template from moodle and fill it with content. Please *only* provide results that are requested in the slides. Upload your presentation before the deadline to moodle.

Exercise 5. (*Ideas for advanced features*)
Have some fun and experiment with the tools you have now mastered. Here are some ideas on what you might want to work on, but feel free to try anything that tickles your fancy.

a) Increase the number of vehicles in the platoon.

b) Have vehicles join and leave the platoon.

c) Have multiple platoons drive, consider collision avoidance between platoons.

d) Have different constraints on agent dynamics drive in the same platoon.

References

[1] Lunze, Jan. Networked Control of Multi-agent Systems: Consensus and Synchronisation, Communication Structure Design, Self-organisation in Networked Systems, Event-triggered Control. Bookmundo, 2019.